DOCUMENTATIE

Tema 3

WAREHOUSE

NUME STUDENT: FLOREAN EUSEBU-IOAN

GRUPA: 30223

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 4](#_Toc95297887)

[4. Implementare 6](#_Toc95297888)

[5. Rezultate **Eroare! Marcaj în document nedefinit.**](#_Toc95297889)

[6. Concluzii 8](#_Toc95297890)

[7. Bibliografie 8](#_Toc95297891)

# Obiectivul temei

Obiectivul acestei teme este crearea unei aplicatii numita “Orders\_management” care permite crearea, editarea, inserarea si stergerea de noi clienti sau produse. De asemenea, se pot crea comenzi care vor verifica valabilitatea produsului, iar in cazul in care acesta este disponibil, va fi create o nota de plata. Pe langa aplicatia in Java cu interfata grafica, este prezenta si o baza de date care permite stocarea tuturor datelor. O bază de date este o colecție organizată de informații sau de date structurate, stocate electronic într-un computer. O bază de date este controlată, de regulă, de un sistem de management al bazelor de date (DBSM). Cumulat, datele, DBMS și aplicațiile asociate reprezintă un sistem de baze de date, denumit prescurtat bază de date.

Datele din cele mai obișnuite tipuri de baze de date sunt distribuite de regulă pe linii și coloane, în diferite tabele, pentru eficientizarea procesării și interogării datelor. Datele pot fi accesate, gestionate, modificate, actualizate, controlate și organizate cu ușurință. Majoritatea bazelor de date utilizează un limbaj structurat de interogare (SQL) pentru scrierea și interogarea datelor.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

In momentul rularii aplicatiei, pe ecran vor aparea 3 ferestre care reprezinta cele 3 frameuri pentru operatiile necesare depozitului de lucru.

Prima fereastra, similara celei din poza alaturata prezinta un tabel, JTable. Capul acestui tabel a fost create in mod dinamic prin intermediul reflexiei. Pentru afisarea datelor s-a create o matrice in care liniile reprezinta inregistrarile, iar coloanele sunt atributele specific fiecarui camp. In partea stanga a tabelului, se gasesc JLabel-uri care ne indica datele ce pot fi introduse, iar dedesubtul acestora avem jtextfield-urile care ne permit inserarea datelor. De asemenea, acest frame contine butoane pentru fiecare operatie, iar la apasarea acestora se va actualiza atat baza de date, cat si tabelul care ne afiseaza in mod lizibil continutul bazei de date. Fiecare client va avea stocat informatii despre el precum primul nume, al doilea nume, adresa si numarul de telefon. In cazul butonului de edit, aceasta operatie se poate face inserand id-ul clientului si modificand oricate campuri se doreste.

O imagine care conține masă

Descriere generată automat

Cea de-a doua fereastra este similara primei, doar ca in acest caz se modifica atributele, acum fiind vorba de produse. Operatiile sunt aceleasi, adica avem posibilitatea de a vizualiza produsele disponibile, de a crea si insera noi produse, de a le edita sau a le sterge. Similar, acest frame contine butoane iar la apasarea acestora se va actualiza atat baza de date, cat si tabelul din interfata grafica. In cazul produselor, informatiile despre acestea se refera la id, nume, pret si stoc disponibil.

O imagine care conține masă

Descriere generată automat

Cea de-a treia fereastra este mai simplista si se ocupa cu gestionarea vanzarii produselor, verificarea stocului si actualizarea datelor, atat in tabel cat si in baza de date. Singura operatie posibila este crearea unei comenzi, insa o data cu aceasta comanda, in spate vor avea loc mai multe alte operatii precum find Client By Id, Find All, create Client, update Client, delete Client, make Bill, etc.

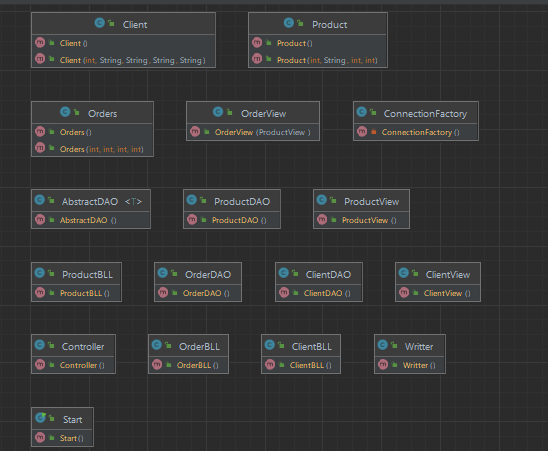
Pentru a putea crea o noua comanda, e nevoie de id-ul comenzii, id-ul clientului care va face comanda, id-ul produsului ce se doreste a fi comandat si cantitatea dorita.



# Proiectare

Aplicatia este creata in InteliJ, iar baza de date in MySqlWorkbench. In Java, codul este structurat in 6 pachete: DAO, care contine legatura dintre baza de date si codul Java, adica mai exact aici se poate observa utilizarea reflexiei. Acest pachet contine 4 clase: AbstractDao= clasa mare cu toate metodele pe caz general, ClientDao, ProductDao, OrderDao= clasele care extind ClasaDao, astfel putand fi aplicate operatiile asupra tuturor obiectelor, indiferent daca tipul lor este Client, Product sau Order. BLL, pachetul care preia operatiile din DAO si permite folosirea acestora in cadrul interfetei grafice. Clasele din acest pachet sunt ClientBll, ProductBll si OrderBll. Un al treilea pachet este Connection cu clasa factoriConnection care face legatura dintre baza de date si Intelij prin intermediul JDBK. JDBC ( Java Database Connectivity ) este o interfață de programare creată de Sun Microsystems - achiziționată de Oracle Corporation - pentru programe care utilizează platforma Java . Permite aplicațiilor Java să acceseze surse de date pentru care există drivere JDBC printr-o interfață comună. În mod normal, aceasta este o bază de date relațională , iar driverele JDBC sunt disponibile pentru toate sistemele de baze de date relaționale cunoscute. Al 4-lea pachet, model contine clasele Client, Product si Order. Fiecare clasa are exact aceleasi atribute cu aceleasi nume ca si cele trei tabele din baza de date, ceea ce faciliteaza conexiunea cu baza de date. Cel de-al 5-lea pachet, presentation contine clasele specifice Jframe care creaza cele 3 ferestre, ClientView, ProductView si OrderView. Jframe este specific Java Swing, o bibliotecă grafică pentru limbajul de programare Java , care face parte din pachetul Java Foundation Classes (JFC), inclus în J2SE . Swing este una dintre principalele modificări aduse de Java 2 comparativ cu versiunile anterioare.Swing oferă posibilitatea de a crea interfețe grafice identice , indiferent de sistemul de operare subiacent, cu prețul unei performanțe mai mici decât folosirea Abstract Window Toolkit (AWT). Folosește principiul Model-View-Controller (MVC, componentele Swing joacă de fapt rolul de vedere în sensul MVC ) și are mai multe opțiuni de aspect pentru fiecare dintre componentele standard.

Odată cu apariția Java 8 în Martie 2014, JavaFX devine biblioteca grafică oficială a limbajului Java, pentru tot felul de aplicații (aplicații mobile, aplicații desktop, aplicații Web), dezvoltarea predecesorului său Swing fiind abandonată (cu excepția remedierilor de erori). Ultimul pachet este “start” si prezinta clasa Start cu metoda statica care permite rularea programului.



Mai sus este ilustrata o imagine care contine toate clasele din aplicatia creata, impreuna cu constructorii lor. Am evitat ilustrarea tuturor variabilelor si ale metodelor deoarece aplicatie este foarte complexa si schema ar fi devenit foarte greu de inteles.

# Implementare

PACHETUL MODEL : CLASELE CLIENT.JAVA PRODUCT.JAVA ORDERS.JAVA

Client.java = atributele :id(int), first Name(String), last Name(String), address(String), phoneNumber(address). Aceasta clasa contine aceste attribute in modul privat, fiind vizibile doar in cadrul acestei clase, asa ca pe langa constructor mai avem si setter si getteri.

Product.java = atributele :id(int), name(String), price(int), stock(int)

Aceasta clasa contine aceste attribute in modul privat, fiind vizibile doar in cadrul acestei clase, asa ca pe langa constructor mai avem si setter si getteri.

Order.java = atributele :id(int), id Client(int),id Product(int), quantity(int).

Aceasta clasa contine aceste atribute in modul privat, fiind vizibile doar in cadrul acestei clase, asa ca pe langa constructor mai avem si setter si getteri.

PACHETUL DAO : CLASELE ABSTRACTDAO.JAVA CLIENTDAO.JAVA PRODUCTDAO.JAVA ORDERDAO.JAVA

AbstractDao.java : private final Class<T> type;

Aceasta clasa contine o constanta private type de tipul T. pe langa constructor, aceasta clasa contine mai multe metode precum:

-createSelectQuery= metoda de tipul String care returneaza query ul in Sql specific afisare din baza de date

-createInsertQuery= metoda de tipul String care returneaza query ul in Sql specific inserarii in baza de date

-createDeleteQuery= metoda de tipul String care returneaza query ul in Sql specific stergerii din baza de date

-createUpdateOrder= metoda de tipul String care returneaza query ul in Sql specific editarii elementelor din baza de date

- void insert = metoda care apeleaza query ul de insert asupra bazei de date si implicit executa operatia de inserare

- void delete = metoda care apeleaza query ul de delete asupra bazei de date si implicit executa operatia de stergere

- void update = metoda care apeleaza query ul de edit asupra bazei de date si implicit executa operatia de editare si actualizare

- void findbyId = metoda care apeleaza query ul de select asupra bazei de date si implicit executa operatia de afisare a unui set de obiecte

- void findAll = metoda care apeleaza query ul de select asupra bazei de date si implicit executa operatia de afisare a tuturor obiectelor din baza de date

-String returntableHead = metoda care returneaza capul de tabel in functie de atributele fiecarui obiect.

ClientDao.java: clasa care mosteneste clasa AbstractDao <Client>, ceea ce permite folosirea tuturor operatiilor definite in clasa parinte.

ProductDao.java: clasa care mosteneste clasa AbstractDao <Product>, ceea ce permite folosirea tuturor operatiilor definite in clasa parinte.

OrderDAO.java: clasa care mosteneste clasa AbstractDao <Order>, ceea ce permite folosirea tuturor operatiilor definite in clasa parinte.

PACHETUL CONNECTION : ConnectionFactory.java

Singura clasa din acest pachet contine conexiunea cu baza de date. Aceasta se face specificand user-ul, parola, driverul si dburl-ul. private static final Logger *LOGGER* = Logger.*getLogger*(ConnectionFactory.class.getName());  
private static final String *DRIVER* = "com.mysql.cj.jdbc.Driver";  
private static final String *DBURL* = "jdbc:mysql://localhost:3306/products\_management";  
private static final String *USER* = "root";  
private static final String *PASS* = "root";

PACHETUL PRESENTATION : ClientView.java, OrderView.java, ProductView.java, Controller.java

Clasa ClientView.java creaza frame-ul care permite executarea tuturor operatiilor in cee ace priveste clientii, precum adaugarea, editarea si stergerea acestora. Aceasta clasa contine o metoda, void show() care va instantia frame-ul, va adauga toate elementele acestuia, urmand ca apoi sa il faca vizibil.Toate aceste evenimente se vor intampla foarte rapid, parand a fi instante pentru ochiul uman.

Clasa Product.java creaza frame-ul care permite executarea tutorur operatiilor care implica organizarea produselor precum inserarea de noi produse, eliminarea acestora sau editarea anumitor date precum stoc, pret, nume sau id.

Clasa Product.java creeaza frame-ul necesar plasarii unei noi comenzi. Acesta contine mai multe campuri text care perimite selectarea unui client, a unui produs, a cantitatii dorite si a unui id\_comanda. Aceasta fereastra este create si afisata dedesubtul celorlalte doua mai sus descrise pentru a facilita accesul la informatii, fiind mult mai usoara alegerea clientilor sau a produselor disponibile.

Clasa Controller.java contine cate o instanta a tuturor acestor clase si rolul ei este de a face legatura intre interfata grafica si tot codul care reprezinta logica aplicatiei.

PACHETUL START: START.JAVA

Singura clasa din acest pachet, Start.java contine metoda statica prin care are loc pornirea executiei aplicatiei. Aceasta executie este realizata prin instantierea unui obiect al clasei Controller.

# Concluzii

In urma dezvoltarii acestei aplicatii numite Orders Management am deprins o multitudine de informatii in ceea ce priveste atat programarea orientate pe obiecte, cat si limbajul de programare Java. Sunt de parere ca este foarte benefic pentru student sa aiba de realizat astfel de proiecte in primul rand pentru ca acestea reprezinta lucruri practice care te dezvolta si te pregatesc pentru cariera in programare si in al doilea rand intrucat aceasta aplicatie reprezinta un proiect personal care poate fi atasat la CV-ul fiecaruia.

De asemenea, implementarea acestei aplicatii m-a ajutat sa inteleg mai bine cum pot crea aplicatii care pot fi folositoare in viata de zi cu zi si a chiar mi-a dat idei de viitoare aplicatii pe care as putea sa le implementez in viitor.( modificarea acestei aplicatii si crearea posibilitatii de a fi descarcata si rulata pe telefon).

Un alt beneficiu al dezvoltarii acestui preiect este reprezentat de faptul ca am reusit mult mai bine sa inteleg idea dintre partea de front-end si back-end. Aceasta aplicatie a fost create folosind ambele strategii. Front-End-ul este partea aplicației care interacționează în mod direct cu utilizatorul. Cunoscută și sub numele de partea client, aceasta se ocupă de aspectul și funcționalitatea interfeței aplicatiei.Elementele implementate de către un Front-End developer aplicației includ, dar nu sunt limitate la: stilul, paleta de culori, imaginile, butoanele, meniurile de navigație, comportamentul elementelor vizibile de către utilizator. Back-End-ul este partea aplicației care nu interacționează în mod direct cu utilizatorul. Cunoscută și sub numele de partea server, aceasta se ocupă de stocarea și managementul datelor prin intermediul bazelor de date, logica de business a aplicației și scrierea de API-uri – interfețe programatice ce expun date din Back-End către aplicația Front-End, și așadar, către client.

Deoarece aplicația Front-End este expusă în mod direct clientului, marea parte a logicii de business a aplicației este scrisă în Back-End, pentru a putea evita accesarea logicii aplicației de către persoane din exterior. Comunicarea cu bazele de date se face exclusiv în Back-End, pentru a ne feri de atacuri de tip malware la nivelul bazei noastre de date.

# Bibliografie

<https://ro.wikipedia.org/wiki/Fir_de_execu%C8%9Bie>

<https://examples.javacodegeeks.com/java-thread-example/>

https://dsrl.eu/courses/pt/

<https://ro.wikipedia.org/wiki/Model-view-controller>